The Large Millimeter Telescope Monitor & Control System

Kamal Souccar, Gary Wallace, and Daniella Malin

Large Millimeter Telescope, Astronomy Department,
University of Massachusetts, Amherst, MA 01003, USA

November 29, 2004

1 INTRODUCTION

The Large Millimeter Telescope, LMT, is composed of a set of devices and instruments that are controlled in a coordinated scheme to perform scientific tasks and collect corresponding data. The status of these devices must be monitored in real-time to assure safety and scientific integrity. This is what typically describes a telescope monitor and control system.

The traditional approach to creating a monitor and control system has been to implement individual device level controllers and then attach them to a client/server system to achieve the desired coordination. The problems in this approach are twofold: complexity and inflexibility. Synchronization and communication protocols must be implemented, and any upgrades or additions require code modification and perhaps even design changes.

Our solution is to automate the creation of a framework for monitor and control by describing the system components in XML and then automatically generating source code for extendible base classes and user interfaces. This enables the monitor and control system to be both flexible and adaptable, and greatly simplifies the design complexity. It also allows the system to be reusable in many different applicable problem domains.

To simplify the coordination mechanism among the different subsystems, we have employed a global state system for the communication model. A single global state object containing references to all of the components of the system is described in XML and corresponding source code is automatically created to implement global object access.

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **21 DEC 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **The Large Millimeter Telescope Monitor & Control System**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Massachusetts, Amherst, MA 01003, USA**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001773, Large Millimeter Telescope Project. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**15**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

2 Object Oriented Design and Automation

2.1 Object Oriented Design

The LMT consists of a large number of complex control systems: the main axes servo system (16 motors in azimuth, 4 motors in elevation), the subreflector positioner (Stewart platform configuration) and wobbler (2 degree-of-freedom system), the active surface panel actuators (720 actuators), and tertiary optics (large flat mirror and several additional smaller mirrors). In addition, the LMT has a collection of scientific instruments such as receivers, spectrometers, backends, and data acquisition systems.

Since the telescope system is naturally comprised of real-world objects, an object oriented design for the monitor and control system logically follows. Each subsystem can be defined as a software object with attributes that describe the properties of the real object and methods that alter the state of these attributes.

2.2 Automation and XML

Due to the complexity of the LMT system, the traditional approach of hand coding the classes describing the objects is both tedious and repetitive. Automation is highly desired to eliminate unnecessary labor and insure a common syntax.

XML[1], the extensible markup language, was the choice for automation. XML is a set of rules for structuring data such as spreadsheets, address books, and configuration parameters. XML simplifies the computer's task of generating and reading data, and ensures that the data structure is unambiguous.

To eliminate the tedious programming of similar class definitions and methods, the properties of each telescope subsystem are specified in an XML configuration file. Each XML file describes an object class including field types and access methods. These XML files are processed to automatically generate extensible base classes and communication methods in C++ and Java; and IDL interface definitions to generate CORBA communication code.

An example of such an XML file is listed below:

```
<config xmlns='http://www.lmtgtm.org'>
  <class>
    <name>Telescope</name>
    <comment>The Telescope class</comment>
  </class>
  <fields>
    <field>
      <name>AzAct</name>
      <type>double</type>
      <set>degreeToRadian</set>
      <get>radianToDegree</get>
      <comment>Actual azimuth angle</comment>
    </field>
    <field>
```

```
      <name>ElAct</name>
      <type>double</type>
      <set>degreeToRadian</set>
      <get>radianToDegree</get>
      <comment>Actual elevation angle</comment>
    </field>
  </fields>
</config>
```

An additional XML file describes the system as a whole by enumerating the instances of each object. This XML file is processed to generate a container object that holds access to all objects in the system.

```
<config xmlns='http://www.lmtgtm.org'>
  <class>
    <name>System</name>
    <comment>The system: contains all objects</comment>
  </class>
  <object>
    <name>TimePlace</name>
    <comment>The system clock and observatory location</comment>
  </object>
  <object>
    <name>Telescope</name>
    <comment>The telescope main axis servo</comment>
  </object>
  <object>
    <name>Sequoia</name>
    <comment>The sequoia frontend</comment>
  </object>
  <object>
    <name>Correlator</name>
    <comment>The correlator backend</comment>
  </object>
  <object>
    <name>Calibration</name>
    <comment>The observing method calibration</comment>
  </object>
  <object>
    <name>Position</name>
    <comment>The observing method position switching</comment>
  </object>
  <object>
    <name>Otf</name>
    <comment>The observing method on-the-fly mapping</comment>
  </object>
</config>
```

The use of XML in astronomy was initiated by work done at NASA's Goddard Space Flight Center[2].

2.3 The LMT XML to Monitor & Control Compiler

The LMT XML to monitor and control compiler (LMT-XMLMC) is a software tool that, given a set of XML configuration files, creates extendible base classes and user interfaces to form a framework for monitor and control. The software was designed around radio telescopes but can be used to monitor and control any system with similar requirements.

The LMT-XMLMC compiler processes the XML configuration files, and generates Java, C++, JNI, and CORBA base classes, as in Figure 1.

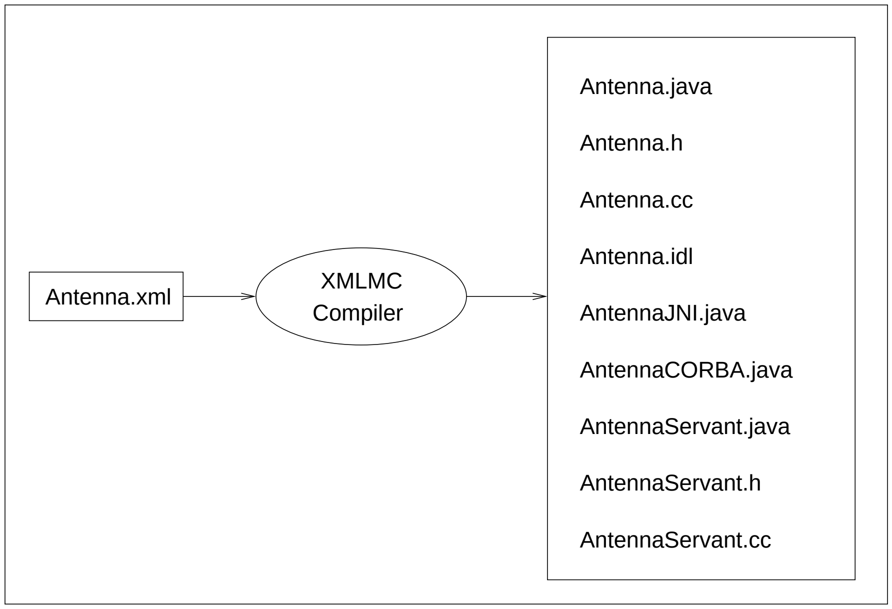


Figure 1: The LMT XMLMC Compiler.

Java [3] is the language of choice for user interfaces. By using the Java Foundation Classes and Swing GUI components, the deployment of user interface applications is greatly simplified and a customizable look and feel is permitted without relying on any specific windowing or operating system.

C/C++ is still the main language to implement device drivers and device controllers. C++ has the advantage of built-in object oriented technology.

JNI[4], the Java Native Interface is the native programming interface for Java, it allows Java code to operate with applications and libraries written in other languages, such as C or C++. The use of JNI is not practical outside of a local area network (LAN) due to bandwidth limitations, hence CORBA.

CORBA [5] is the acronym for Common Object Request Broker Architecture. It is a vendor-independent architecture and infrastructure that computer applications can use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network.

Therefore, the Java base classes are used to drive the Java user interface, the C++ bases classes drive the real-time system device drivers and controllers, the JNI classes enable communication between the Java and C++ classes on the LAN, and the CORBA classes enable communication between the Java and C++ classes among any computers.

In addition, the LMT-XMLMC compiler creates a single object in Java and C++ that contains references to all of the elements of the system to implement a global state system. Any object can read the state of any other object by acquiring a reference to that object from the global state object.

It is important to note that the LMT-XMLMC compiler does not generate device driver or controller code, but provides for extendible base classes that share a common interface and provide a basis for further development.

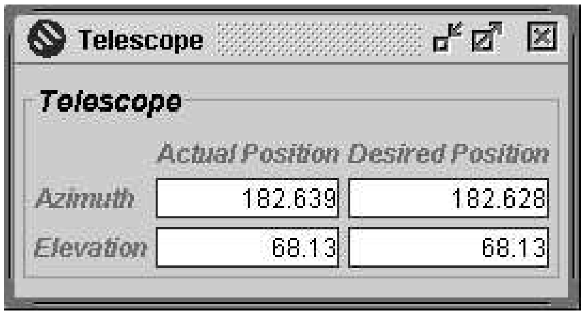
2.4 Automation and User Interface XML

To guarantee a consistent look and feel to the LMT monitor and control system, a set of XML configuration files are used to describe the user interface and layout of the monitor and control panels. One XML file defines each panel as a *Monitor*, *Control*, or *MonitorControl* panel. Java Swing code is generated from these XML files at run time to layout the panels and display the system status and allow for user input.

An example of a UI XML file follows.

```
<Panel xmlns='http://www.lmtgtm.org'>
  <Tab/>
  <Label>
    <value>Actual Position</value><align>Center</align>
  </Label>
  <Label>
    <value>Desired Position</value><align>Center</align>
  </Label>
  <Return/>
  <Label>
    <value>Azimuth</value>
  </Label>
  <Text>
    <field>AzAct</field><align>Right</align>
  </Text>
  <Text>
    <field>AzDes</field><align>Right</align>
  </Text>
  <Return/>
  <Label>
    <value>Elevation</value>
  </Label>
  <Text>
    <field>ElAct</field><align>Right</align>
  </Text>
  <Text>
    <field>ElDes</field><align>Right</align>
  </Text>
  <Return/>
</Panel>
```

This results in the following panel:



	Actual Position	Desired Position
Azimuth	182.639	182.628
Elevation	68.13	68.13

Figure 2: An Example LMT Monitor & Control Panel.

3 Global State System

As mentioned above, a global state system was used to facilitate access among the different system components and simplify the communication protocol.

3.1 Global State System Motivation

The common approach to telescope control is to define communication paths between the different subsystems and implement a message/data passing scheme to achieve the desired coordination.

An example scenario is presented in Figure 3. The arrows indicate communication channels between servo systems and instruments, and control modules. The picture illustrates the complexity of the problem. For instance, as the beam is tracking the source, data are being collected and recorded, and the behavior of the system is being monitored. The source tracker effectively controls the position of the antenna, the subreflector, the active surface, and requires information about the environment temperature, humidity, and wind speed. Concurrently, the data acquisition system controls the activation of the instruments. It also records data and tags each sample with the exact position of the beam and the environment conditions. Finally, as the source is being tracked and data is being recorded, the condition of the system is being overseen by the monitoring system.

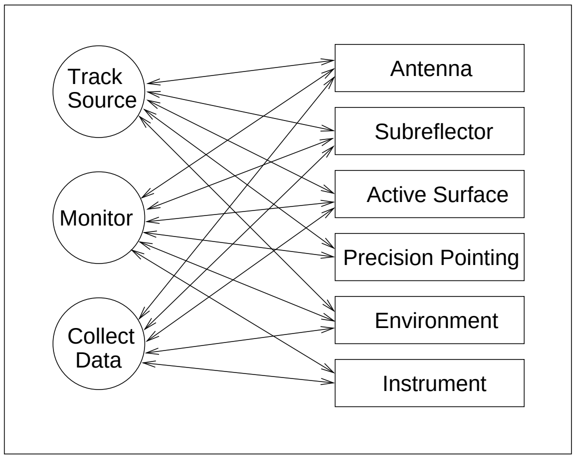


Figure 3: A Typical Information Exchange Between Telescope Subsystems.

The approach we proposed is a global state system. Each subsystem posts its state to the global state and retrieves the state of other subsystems from the same global state. A finite state machine controller coordinates the activities between the different subsystems through that same global state. This approach is illustrated in Figure 4.

The only requirement to make this approach feasible is that for each element in the global state, only one writer can exist. This ensures that two or more processes cannot simultaneously write or update a given element. On the other hand, an unlimited number of readers can access the same global state element.

3.2 Implementation of the Global State System

3.2.1 Shared Memory

On a single computer system, the global state space can be implemented using shared memory – each object is assigned a unique key, therefore a unique memory space. In a distributed computing system, a replicated shared memory system is used.

Replicated shared memory is implemented by installing an individual memory board in each computing system and interconnecting these memory boards using a fiber optic link. Memory writes to a replicated shared memory board at one computer are instantly sent to all other replicated shared memories on the network.

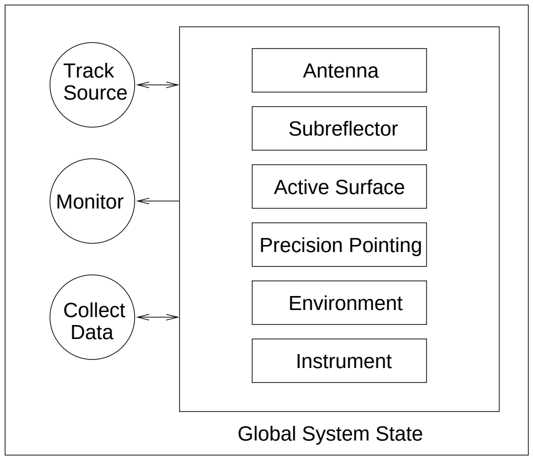


Figure 4: Global State Approach.

3.2.2 CORBA

When direct access to the replicated shared memory is not available, as in remote observing, a CORBA server is used to provide access to the shared memory. The CORBA server is automatically generated using the LMT-XMLMC compiler and provides access to all of the objects in the system.

ACE/TAO [6] and JacORB [7] are used in the implementation of the CORBA server.

The idea to use CORBA for the control system of the LMT is based on GTC's control system [8].

3.3 Computing Environment

Choosing the correct computing environment for the LMT was a difficult task. The final choice was to use a host/target system: a Sun workstation running Solaris, and a VME-based Motorola PowerPC embedded computer running VxWorks. This configuration was chosen for the following reasons:

- Sun Microsystems is a leader in the field of vendor supported Unix-based computers; the hardware failure rates are minimal and security and bug support for Solaris are readily available from Sun. The use of Linux on an Intel platform was investigated, however, due to the continuous effort in keeping Linux up-to-date and the occasional lack of backward compatibility, it was decided to pay the additional cost to purchase and use vendor-supported software and hardware rather than provide that support in-house. This enabled the concentration on developing the application and saved time with respect to supporting the development environment

- The VME bus remains a powerful and stable bus system for control applications. While the cost of VME-based devices is higher than other alternatives, the expandability of such systems and the availability of a wide choice of third party products for the VME system makes the additional cost more easily justified.
- Motorola PowerPC embedded computers were chosen due to the continuity of the Motorola embedded computers product line. The expected life-time of the LMT is 30 years and Motorola is committed to providing consistency in the upgrade path of their processors.
- VxWorks is one of the market leaders in real-time operation systems. While the purchase cost for VxWorks remains high, the direct mapping between Unix and VxWorks and the wide availability of device drivers in that environment justifies the initial cost.

4 The Large Millimeter Telescope Monitor & Control System

The system described so far is used to build the LMT monitor and control system (LMTMC). The LMTMC, Figure 5, is composed of several modules including an observing tool, monitoring tools, a finite state machine controller, and a scheduler.

4.1 Observing Tool

The observing tool provides the means for the user to create observing programs. These observing programs can then be executed on-line or submitted to the scheduler for optimal execution. They can also be saved to or loaded from a file, and can be manually edited using any text editor.

Each observing program must contain scheduling constraints, target positions, a receiver, a backend instrument, and a data collection method.

In turn, the observing tool generates commands to control the different subsystems based on the observing program. The generated commands are implemented using a keyword-value pair method that manipulates the parameters of each subsystem.

The observing tool can also be used to deliver commands directly to the system in a more interactive manner for both scientific and engineering purposes.

The observing tool provides the following scientific functions:

- source definition: (az/el; ra/dec; l/b; ephemeris)
- pointing offsets
- radiometer control: selecting a detector, changing frequency, doppler tracking
- spectrometer control: selecting a spectrometer, multiple IFs, bandwidth, resolution
- wobbler control: frequency, magnitude
- observing mode: pointing system modeling, calibration, position switching, frequency switching, beam switching, five point mapping, pointed maps, scanned maps, on-the-fly mapping

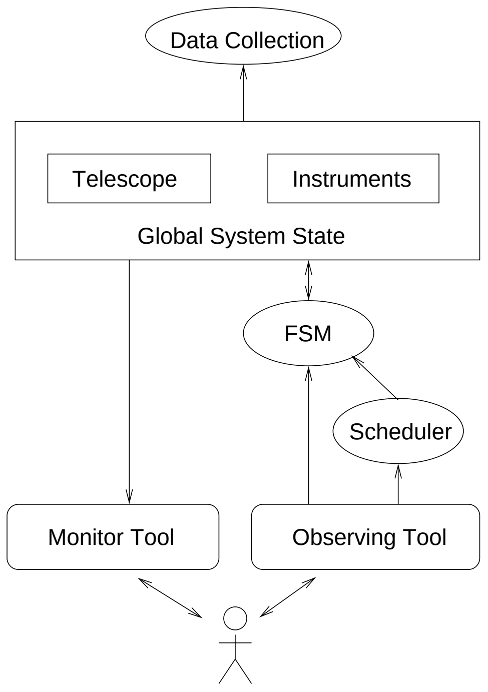


Figure 5: The LMT Monitor & Control System.

- integration time

In addition to the scientific functions, a set of engineering functions can be accessed from the observing tool. These engineering functions are the building blocks for the scientific functions.

- antenna azimuth and elevation control
- subreflector control
- active panel control
- tertiary optics control
- instrument control

The following resources are made available to the user:

- online help
- source catalogs
- line catalogs
- previewers (ex. map configuration)
- estimates of time overheads
- weather conditions: atmospheric opacity over previous 12 hours, wind speed and direction, phase monitor

Many of the ideas including the look of the observing tool were inspired by Gemini's observing tool [9].

4.2 Scheduler

Once observation plans are defined, they get submitted to the scheduler. The plans are thereafter scheduled based on two different policies:

- dynamic scheduling: observations are scheduled based on meteorological conditions and scientific priorities.
- manual scheduling: observation times are assigned manually to observers.

4.3 FSM Controller

A finite state machine controller translates the observing programs into desired telescope and instrument states. It steps through the science program and executes each command while monitoring the error state of the system.

4.4 Monitoring Tool

The monitoring tool is used to oversee the state of the system in real-time while observing commands are being executed, and scientific and engineering data are being collected.

The purpose of the monitoring tool is to provide a running check on data integrity and proper system operation by displaying the state of the system to the user.

The following information can be displayed using the monitoring tool:

- observation state: schedule step, elapsed time, time to completion, ...
- telescope state: position, temperature, ...
- instrument state: current configuration, bandwidth, sampling rate, ...
- current pointing model
- active surface state
- subreflector state
- weather conditions
- error signals
- logs

The concepts for the monitoring tool were based on OVRO's Java windows [10].

5 Adapting to Existing Telescopes

To prepare for launching the monitor and control system on the LMT, several systems were built or upgraded to use the LMTMC architecture. These diverse systems demonstrate the reusability and adaptability of the LMTMC framework.

5.1 LMT Simulator

As a first approach to verify the operation of the LMTMC system, a simulator of the LMT was developed. The simulator assumes the behavior of the real telescope and instruments and provides a teaching ground for future users of the system.

5.2 IOTA

IOTA [11], the Infrared Optical Telescope Array on Mount Hopkins in Arizona is a three element optical interferometer with four delay lines, a three-beam combiner, and a complex data acquisition system. The control system was a non-networked collection of Apple Macintosh computers. Each Mac controlled a different device of the system and it was up to the observer to coordinate the operation of the devices. The old IOTA system was replaced by a Sun-Solaris/VME-PowerPC-VxWorks system running the LMTMC software as part of a general IOTA upgrade [11, 12]. This upgrade provided a proving ground for the LMTMC system before launching on the LMT. A single VME cage with three embedded PowerPC processors runs all three telescopes, the delay lines, and all of the instruments. The IOTA system has been in operation for three years.

5.3 FCRAO Narrowband Spectrometer

The LMTMC software was used to develop the control architecture of the Five College Radio Astronomy Observatory narrowband spectrometer. It is a spectrometer with 64 input channels, 1024 lags per channel, with on-the-fly mapping capability.

5.4 FCRAO 14m Telescope

The FCRAO is the Five College Radio Astronomy Observatory which consists of a 14 meter radio telescope enclosed in a radome [13]. The FCRAO control system was a Modcomp computer networked using a special purpose parallel I/O to ethernet bridge. This system uses the same instruments that will be used at the LMT and is a facility run by the University of Massachusetts. The FCRAO control system was replaced by a Sun-Solaris/VME-PowerPC-VxWorks system running the LMTMC software and the resulting system will be directly used on the LMT.

6 Conclusion

Modern observatory telescopes are complex distributed systems. We have devised a reusable, automatically generated software system that simplifies the implementation of monitor and control systems for such telescopes. The described system takes advantage of advances in computing technology to facilitate the development of control software, ease the integration of new devices and instruments, and shorten the path to scientific production.

References

[1] “Extensible Markup Language (XML),” in *http://www.w3c.org/XML*, World Wide Web Consortium, 2002.

[2] T. Ames, L. Koons, K. Sall, and C. Warsaw, “Using XML and Java for telescope and instrumentation control,” in *Proc. SPIE Vol. 4009, p. 2-12, Advanced Telescope and Instrumentation Control Software, Hilton Lewis; Ed.*, **4009**, pp. 2–12, June 2000.

[3] “Java,” in *http://java.sun.com*, Sun Microsystems, 2002.

[4] “JNI - Java Native Interface,” in *http://java.sun.com/products/jdk/1.1/docs/guide/jni*, Sun Microsystems, 1998.

[5] “CORBA,” in *http://www.corba.org*, Object Management Group, 2002.

[6] D. C. Schmidt, “Real-time CORBA with TAO (The ACE ORB),” in *http://www.cs.wustl.edu/˜schmidt/TAO.html*, Washington University.

[7] “JacORB,” in *http://www.jacorb.org/*, Freie Universität, Berlin.

[8] R. Penataro, J. M. Filgueira, P. Gomez-Cambronero, M. Gonzalez, and M. Puig, “Application of CORBA to the GTC control system,” in *Proc. SPIE Vol. 4009, p. 152-166, Advanced Telescope and Instrumentation Control Software, Hilton Lewis; Ed.*, **4009**, pp. 152–166, June 2000.

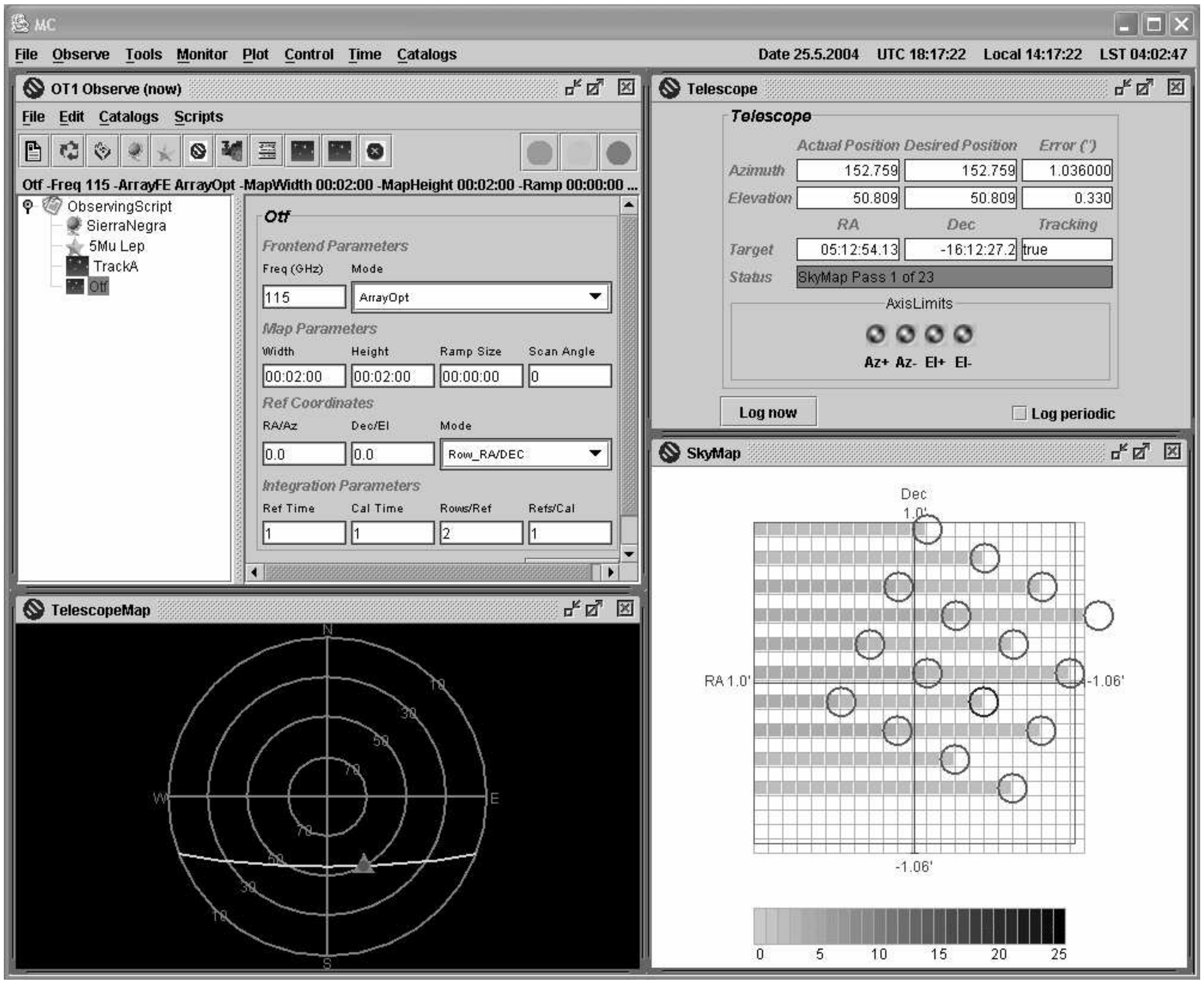
[9] S. Wampler, K. K. Gillies, P. J. Puxley, and S. Walker, “Science planning for the Gemini 8-m telescopes,” in *Proc. SPIE Vol. 3112, p. 246-253, Telescope Control Systems II, Hilton Lewis; Ed.*, **3112**, pp. 246–253, Sept. 1997.

[10] S. Scott and R. Finch, “The New User Interface for the OVRO Millimeter Array,” in *ASP Conf. Ser. 145: Astronomical Data Analysis Software and Systems VII*, **7**, pp. 49–+, 1998.

[11] W. Traub and et. al., “Third telescope project at the IOTA interferometer,” in *Proc. SPIE Vol. 4006, p. 715-722*, pp. 715–722, 2000.

[12] F. Schloerb and et. al., “Control system software for the IOTA,” in *New Frontiers in Stellar Interferometry, Proceedings of SPIE Vol. 5491*, 2004.

[13] “Five College Radio Astronomy Observatory,” in *http://www.astro.umass.edu/fcrao*, University of Massachusetts Amherst, 1996.



Date 25.5.2004 UTC 18:17:22 Local 14:17:22 LST 04:02:47

Otf -Freq 115 -ArrayFE ArrayOpt -MapWidth 00:02:00 -MapHeight 00:02:00 -Ramp 00:00:00 ...

Otf

Frontend Parameters

Freq (GHz)	Mode
115	ArrayOpt

Map Parameters

Width	Height	Ramp Size	Scan Angle
00:02:00	00:02:00	00:00:00	0

Ref Coordinates

RA/Az	Dec/El	Mode
0.0	0.0	Row_RA/DEC

Integration Parameters

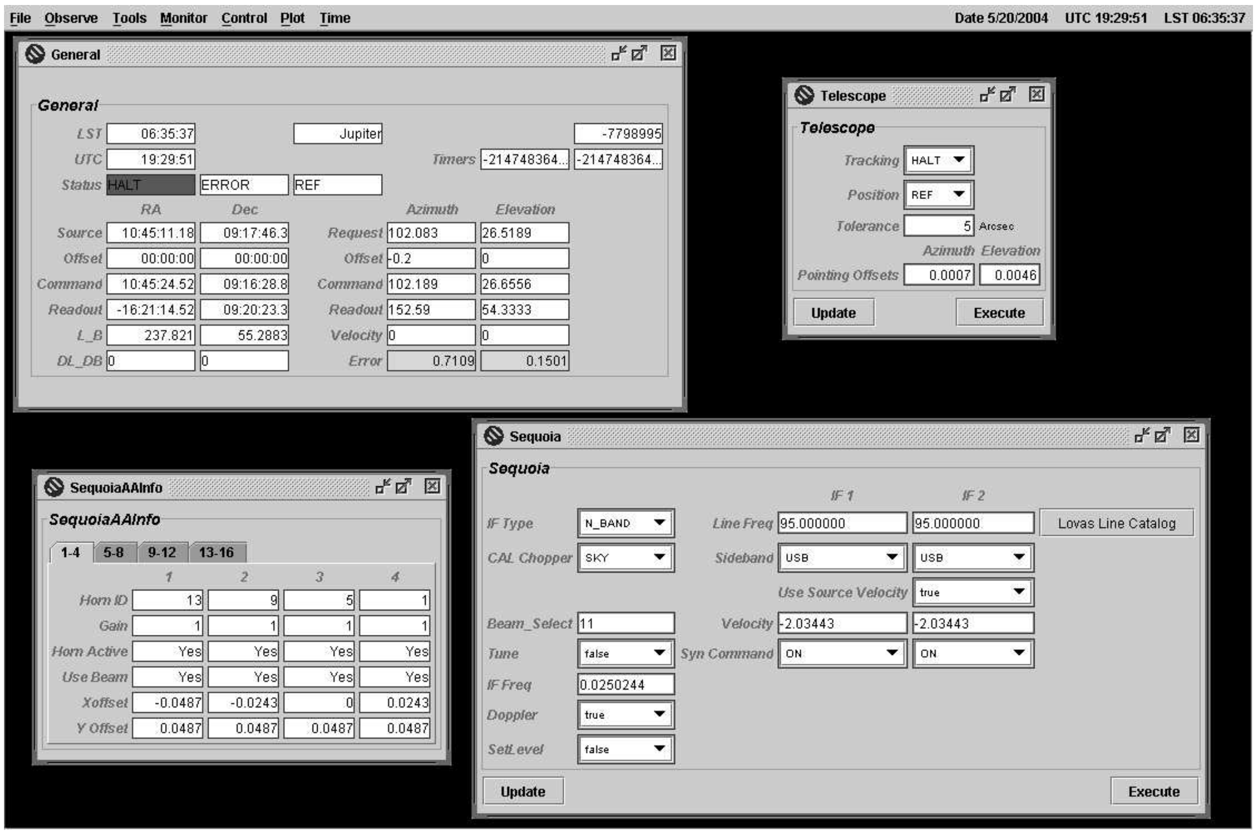
Ref Time	Cal Time	Rows/Ref	Refs/Cal
1	1	2	1

Telescope

	Actual Position	Desired Position	Error (")
Azimuth	152.759	152.759	1.036000
Elevation	50.809	50.809	0.330

	RA	Dec	Tracking
Target	05:12:54.13	-16:12:27.2	true
Status	SkyMap Pass 1 of 23		

Figure 6: The LMT Monitor & Control System.



Date 5/20/2004 UTC 19:29:51 LST 06:35:37

General

LST	06:35:37		Jupiter			-7798995
UTC	19:29:51			Timers	-214748364...	-214748364...
Status	HALT	ERROR	REF			
	RA	Dec		Azimuth	Elevation	
Source	10:45:11.18	09:17:46.3	Request	102.083	26.5189	
Offset	00:00:00	00:00:00	Offset	-0.2	0	
Command	10:45:24.52	09:16:28.8	Command	102.189	26.6556	
Readout	-16:21:14.52	09:20:23.3	Readout	152.59	54.3333	
L_B	237.821	55.2883	Velocity	0	0	
DL_DB	0	0	Error	0.7109	0.1501	

Telescope

Tracking	HALT	
Position	REF	
Tolerance	5	Arcsec
	Azimuth	Elevation
Pointing Offsets	0.0007	0.0046

SequoiaAAInfo

1-4 5-8 9-12 13-16

	1	2	3	4
Horn ID	13	9	5	1
Gain	1	1	1	1
Horn Active	Yes	Yes	Yes	Yes
Use Beam	Yes	Yes	Yes	Yes
Xoffset	-0.0487	-0.0243	0	0.0243
Y Offset	0.0487	0.0487	0.0487	0.0487

Sequoia

			IF 1	IF 2
IF Type	N_BAND	Line Freq	95.000000	95.000000
CAL Chopper	SKY	Sideband	USB	USB
		Use Source Velocity		true
Beam_Select	11	Velocity	-2.03443	-2.03443
Tune	false	Syn Command	ON	ON
IF Freq	0.0250244			
Doppler	true			
SetLevel	false			

Figure 7: Monitor & Control Panels of the FCRAO Telescope.